A NEW AUTOMATED WAY TO DETERMINE RATES FOR LENDING PROTOCOLS WITH LASA - LENDING AUTOMATED SCALING ALGORITHM

CHRISTOPH KRPOUN† AND RENÉ HOCHMUTH

OCTOBER 21, 2022

Abstract. Liquidity pools of fungible tokens are the foundational building blocks of any decentralized lending protocol (e.g. Aave, Venus, etc.) Lately, even NFT lending protocols are utilizing liquidity pools (BendDAO, LiquidNFTs). What these pools all have in common (aside from LiquidNFTs) is that they use a *static* relationship between utilization and rates which determine the lending and borrow annual percentage per year [APY]. The following article introduces a new *dynamic* way to adjust these utilization curves with a one dimensional monte carlo method. The goal is to create a system that is sensitive to macro economic changes which mostly are reflected in user behavior.

Contents

1. Introduction

The state of the art for lending protocols use static rate curves with two different slopes. Sometimes there is even a constant borrow APY with a fixed value for all percentages of utilization. An example is shown in figure 1 for a ETH pool from Aave with a variable rate. Two things are peculiar in this instance. Firstly, there is a point at which the slope of the curve changes rapidly. Thus, the curve still continues, but is sharp and angular rather than smooth.

Secondly, one can ask: how this point is chosen? Is it from data, or arbitrarily defined? If it is from past data, why should it remain the same for all future instances? The point at which the slopes change is often called the optimal utilization U_{opt} and

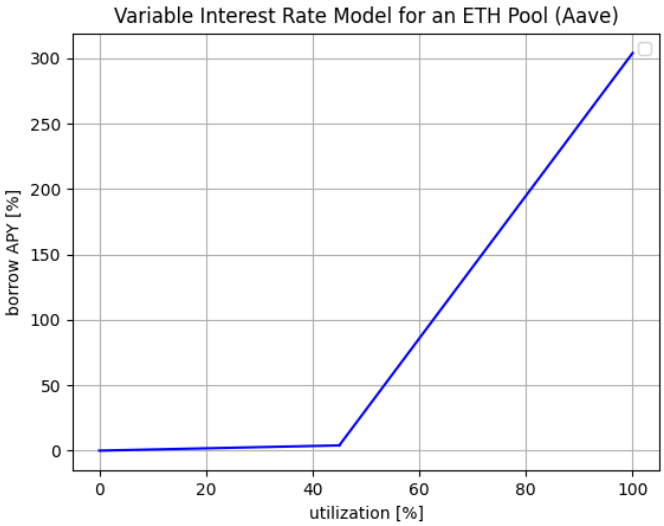

FIGURE 1. Example for a borrow rate - utilization curve. This curve is exactly used in Aave's ETH pools.

should reflect the optimal utilization of an asset type which should be reached. Realistically though, this value isn't static. It can change from time to time, depending on global market and local social conditions.

The goal of the *Lending Automated Scaling Algorithm*, or short LASA, is to get rid of this arbitrariness and define a way to get a smooth bonding curve with a variable U_{opt} for each pool, which mirrors macro economic states. Doing so, this article is separated in the following sections. In the first section we set the theoretical with definitions for LASA and the used assumptions for the simulations. In the second section we talk about the theoretical aspects of LASA and the corresponding bonding curve. Afterwards, we explain the numerical implementation in solidity. In the fourth section we firstly explain the set up for our test environment in which we simulated the behavior of LASA in a *"as realistic as possible"* framework. Secondly, we give numerical results of these simulations. In the last section we sum up the results and state an objective conclusion.

2. PRELIMINARIES

We assume that the reader has a proper mathematical background. Hence, we use the standard terminology used in mathematics and build up a logical setup with definitions and, if necessary, lemmata or theorems. For reader with a lower understanding of mathematics we like to suggest follow up articles which summarize the results for a broader community on medium or gitbook.

We will use the variable A for amounts and specify it with a subscript. Same we will do for the share amounts which we denote by S.

2.1. APY as a Personal Risk Level for Smart Contract Interactions.

Beside telling someone how much they earn on a given principal, APY's tell you something about the risk level of your investment. Normally, a product gives the user higher APY's, when there is a higher risk to loose value of the principal. A good example are

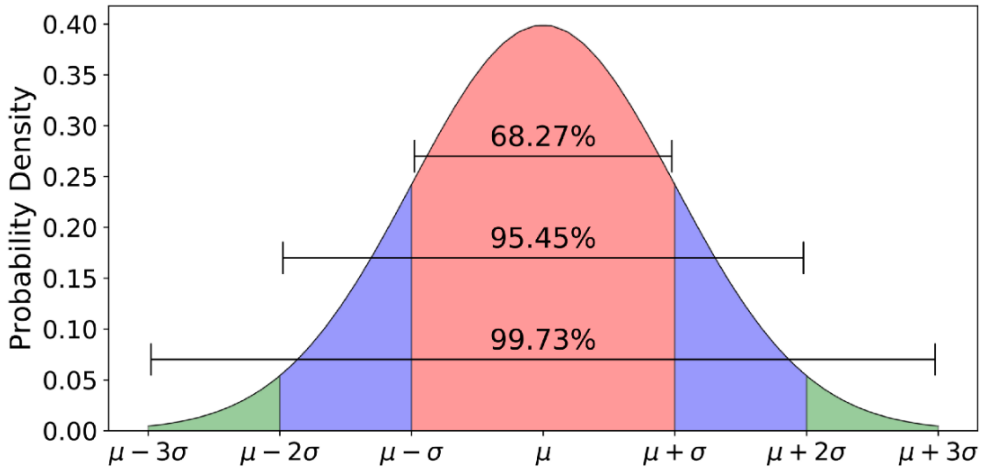

FIGURE 2. Example of a normal distribution. μ: mean value, σ: standard deviation

yield farming rewards for liquidity mining. Because, it is a common thing that new *decentralized exchanges* [**DEX's**] also introduces an own DEX token, they normally provide some farms in which they pair this DEX token with some blue chips. These farms have very high APY's to attract users, because, there is a high chance that the value of the DEX token decreases rapidly. In other words, there is a high risk to loose value from the principal.

Since risk management is a personal trait, it differs from person to person. There are many external influences that can change it on a daily bases. Thus we make following definitions:

Definition 2.1. *A user* $\boldsymbol{u}$ *is an entity which describes a human with personal traits who is interacting with smart contracts. We call* M *the set of all such users with* $u \in M$*.*

Definition 2.2. *Every user has a personal risk* APY *for lending* ξ *and borrowing* η*, displaying their current risk management behavior. Both values can change on a daily bases and are influenced by several external factors.*

It is not possible nor necessary to determine ξ and η for each user individual. Since, it is a personal trait and with that a result of nature. A normal distribution, also known as a Gaus bell shaped curve [see Figure 2][1], can fit as a good approximation. We will later use this distribution in our simulations to get a reasonable setup to test the algorithm.

When we think about the ranges from ξ and η in a real application, we notice that these will never be less then zero. When we compare this with the shape of Figure 2, we notice that we need to truncate the normal distribution.

[1]Picture taken from: `https://www.isixsigma.com/dictionary/normal-distribution/`

Definition 2.3. *Let X be a normal distribution with mean μ and variance σ^2. Also, let the distribution be bounded in the interval $[a, b]$ with $\infty \leqslant a < b \leqslant \infty$. Then X has a truncated normal distribution. Its probability density function P for $a \leqslant x \leqslant b$ is given by*

$$P(x;\, \mu,\, \sigma,\, a,\, b) = \frac{1}{\sigma}\frac{\phi\left(\frac{x-\mu}{\sigma}\right)}{\Phi\left(\frac{b-\mu}{\sigma}\right) - \Phi\left(\frac{a-\mu}{\sigma}\right)} \tag{2.1}$$

with $\phi : [a, b] \longrightarrow [0, 1]$ being the probability density function of the standard normal distribution and $\Phi : [a, b] \longrightarrow [0, 1]$ the cumulative distribution function defined like

$$x \mapsto \frac{1}{2}\left(1 + \operatorname{ref}\left(\frac{x}{\sqrt{2}}\right)\right).$$

By definition we set $\Phi\left(\frac{b-\mu}{\sigma}\right) = 1$ for $b = \infty$ and $\Phi\left(\frac{a-\mu}{\sigma}\right) = 0$ for $a = -\infty$.

We will use this special type of normal distribution to set the allocation of personal risk APY's in the group of users.

Definition 2.4. *The distribution of ξ and η in the set of users M is given by a truncated normal distribution $\mathfrak{T}(M)$* [see definition 2.3]*. We call the tuple $(M, \mathfrak{T}(M))$ a pre-society and $\omega(u)$ the wish rate of an individual user. We separate between lender wish rate $\omega_l(u)$ and borrower wish rate $\omega_b(u)$. Both rates are finite by definition.*

2.2. **The Pareto Distribution to set Lending and Borrowing Amounts.** For later purposes we must set reasonable borrow and lending amounts for our simulations, however, it is not possible to set these individually when working with huge sample sizes. Moreover, like for the APY's, we will choose a certain distribution which can be used to determine the borrow and lending amount of a user randomly. A very good candidate for this is the so called Pareto distribution. A plot of this distribution can be seen in Figure 3 [2]. It was designed to describe the distribution of wealth in a society. Since rich users can lend or borrow more than poor users, this distribution is a good fit for our purposes. Therefore, we make following definition:

Definition 2.5. *Let X be a random variable. We say that X has a Pareto distribution when the probability that X is greater than some number x is given by*

$$P(X > x) = \begin{cases} \left(\frac{x_m}{x}\right)^{\alpha}, & \text{for} \quad x \geqslant x_m \\ 1, & \text{for} \quad x < x_m \end{cases} \tag{2.2}$$

is the minimal value of X with $x_m > 0$. We set $\alpha > 1$ which is called the shape parameter. The corresponding probability density function has the form

$$\phi : \mathbb{R} \longrightarrow [0, \alpha], \quad x \mapsto \begin{cases} \dfrac{\alpha x_m^{\alpha}}{x^{\alpha+1}}, & \text{for} \quad x \geqslant x_m \\ 0, & \text{for} \quad x < x_m \end{cases} \tag{2.3}$$

Definition 2.6. *The borrow and lending amount distribution within the set of users M is described by the pareto distribution $\mathfrak{p}(M)$ from Definition 2.5. We call the triple $(M, \mathfrak{T}(M), \mathfrak{p}(M))$ a society.*

We will use this society set as a basis for our later numerical simulations.

[2] Picture taken from: `https://en.wikipedia.org/wiki/Pareto_distribution`

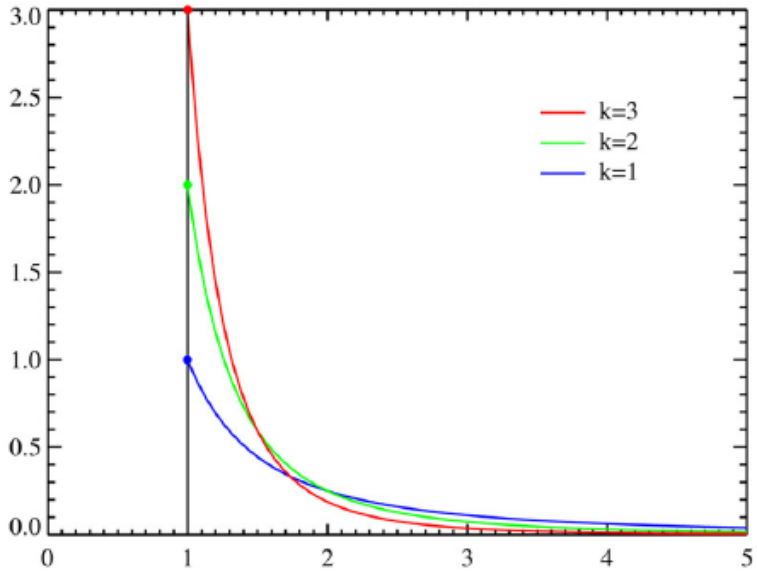

FIGURE 3. Example of a standard Pareto distribution for different shape parameters. $x_m = 1$: minimal value and $\alpha = k$: shape parameter.

3. Setting up LASA - Lending automated Scaling Algorithm

We start by defining a function for the rates which we denoted with R. Doing so, we are setting the *domain of definition* of R, namely $\mathcal{D}(R)$.

Definition 3.1. *The domain of definition for R is defined like*

$$\mathcal{D}(R) = \{\, x \,|\, x \in [0,1] \,\} \tag{3.1}$$

Also, we define the utilization in terms of the total pool amount A_t and pseudo total token amount A_p. Both variables are defined as following:

Definition 3.2. *For each pool we have the variables:*

(i) *The total amount A_t is the number of token being inside the corresponding pool.*

(ii) *The pseudo total token amount A_p ist the total amount plus all gathered interest left in the pool so far. Consequently, we always have $A_p \geqslant A_t$.*

Now, we can define the utilization.

Definition 3.3. *The utilization is a function $U : \mathbb{R}_+ \times \mathbb{R}_+ \longrightarrow [0,1]$ with*

$$(A_t, A_p) \mapsto 1 - \frac{A_t}{A_p}. \tag{3.2}$$

We want to highlight when we set U as a function of A_p, then $U(A_p)$ is bijectiv. Since, it is clearly smooth it is a diffeomorphism.

The goal is to mimic the behavior of Figure 1 but in a smooth way. Thus, we need a function which rapidly increases when reaching a certain value. One class of functions with such a property are the asymptotic functions with a pole at a position p. We normalized it that $R(0) = 0$.

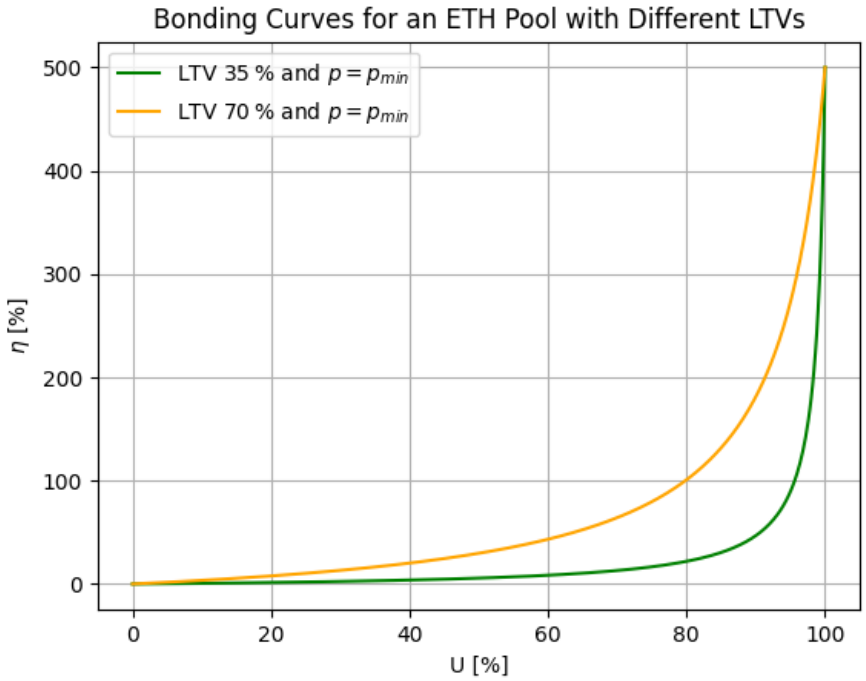

FIGURE 4. Plot of $R(x)$ for different a values.

Definition 3.4. *The relation between borrow rate and utilization - also called bonding curve - is described by the function* $R : \mathcal{D}(R) \longrightarrow \mathbb{R}_+$ *of the form*

$$R(x) := \frac{ax}{(p-x)p} \tag{3.3}$$

with $x \in \mathcal{D}(R)$ *called the utilization,* $p \in \mathbb{R}_+$ *the pole and* $a \in \mathbb{R}_+$ *the scaling factor.*

Later, by choosing boundaries for p we can show that the above relation is smooth and well defined for all $a \in \mathbb{R}_+$. The scaling factor a is a free parameter and used to gauge the corresponding curve to individual pool conditions.

The last and most important parameter is the pole p. This parameter will be changed by LASA within a predefined, closed interval. Thus, LASA shifts the pole in a certain interval resulting in a later or earlier rapid increase of the borrow rate. Therefore, one can imagine p as a new U_{opt} which is always greater than 1 and not static. This solves both problems which we addressed in the introduction. A sketch of a borrow rate curve with different a and $p = p_{min}$ - we will shortly define p_{min} and p_{max} - can be seen in Figure 4.
We want to highlight that by changing the value of p, the curve moves most in the last end near the pole - compare (6). In the following paragraph we want to define the boundaries for p and show that R is a smooth and well-defined function.

Definition 3.5. *p is a bounded parameter within the interval $[p_{min}, p_{max}]$ where we define*

$$p_{min} := \frac{1}{2} + \sqrt{\frac{1}{4} + \frac{a}{1.5}} \quad \textit{and} \quad p_{max} := \frac{1}{2} + \sqrt{\frac{1}{4} + \frac{a}{5}}. \tag{3.4}$$

The values 1.5 *and* 5 *correspond the borrow rates of* 150 % *and* 500 % *at* $U = 1$. *In other words we define the boundaries to a given multiplication factor* $\boldsymbol{a}$ *by saying the maximal borrow rate should be* 150 % *or* 500 % *for* $x = 1$.

We will restrict the range of R to each such maximal value belonging to a p. Now, we have everything together for following theorem.

Proposition 3.6. *With* $p \in [p_{min}, p_{max}]$ *the function* R *from Definition 3.4 is well-defined and* $R \in C^\infty(\mathcal{D}(R), \mathbb{R}_+)$ *for all* $a \in \mathbb{R}_+$. *Additionally, it is bijectiv for each* p *and thus a function family of diffeomorphism* $(R_p)_{p\in[p_{min},p_{max}]}$.

Proof. Since, R is a polynomial in x it is smooth when it is well-defined. Thus we only focus on this part. To show it is well-defined, we need to show that the difference in the denominator of R is never zero. Because, $x \in \mathcal{D}(R)$, it is sufficient to show that $p > 1, \forall a > 0$. Additionally, the interval of p is bounded with $p_{min} < p_{max}$ we we just need to show that $p_{min} > 0$, $\forall a > 0$. With Definition 3.5 we can make following estimate:

$$p_{min} = \frac{1}{2} + \sqrt{\frac{1}{4} + \frac{a}{1.5}} > \frac{1}{2} + \sqrt{\frac{1}{4}} \geqslant 1 \tag{3.5}$$

The first estimate was possible because $a > 0$. The injectivity for each p is clear from the definition of R, and the range of R restricted to the maximum value for each p at $x = 1$ which makes it surjectiv and hence bijectiv. This ends the proof. □

3.1. **Defining Efficiency of a Smart Contract.** As explained in the last subsection, we want to vary the pole parameter p to test for the current optimal utilization of the pool. Doing so, we give some boundaries which are set by the maximal borrow rate and minimal borrow rate at 100 % utilization. We assume that we have a society set $(M, \mathfrak{T}(M), \mathfrak{p}(M))$ and need to find a feedback mechanism which can not be easily attacked. In other words, we want to know if a change of p is "good" or "bad" in a safe way. Since we work with a pareto distribution for amounts, this can only be achieved by looking at longer time. Only, in this case we can talk reasonable about a mean value from the pareto distribution for the user amounts which is given by

$$\mu_1(X) = \frac{\alpha x_m}{\alpha - 1}$$

and exists because $\alpha > 1$ by definition.

In the end, one wants as many users as possible using the contract. As we will show, this can be mirrored by maximizing the lending S_l or borrow shares S_b equivalently when looking at long time frames. We start with defining lending and borrow shares.

Definition 3.7. *Both values are defined like:*

(i) S_l *is the sum of all existing lending shares of the contract. Lending shares describe the portion of the total value of the contract belonging to a user.*

(ii) S_b *is the sum of all existing borrow shares of the contract. Borrow shares describe the portion of the total borrowed amount from a contract belonging to a user.*

For later purpose we define following operators:

Definition 3.8. *Let* $(M, \mathfrak{T}(M), \mathfrak{p}(M))$ *be a society and* $M_l \subset M$ *the subset of lenders and* $M_b \subset M$ *the subset of borrower. We define the projections* $\pi_M^{l,b} : \mathbb{R}_+ \times M \longrightarrow M_{l,b}$

onto those subsets to a givin rate r as

$$\pi^l_M(r) := \{u \in M \,|\, \omega_l(u) \leqslant r\} \quad \textit{and} \quad \pi^b_M(r) := \{u \in M \,|\, \omega_l(u) \geqslant r\}. \tag{3.6}$$

We set the maximal and minimal wish rate in both cases by

$$\omega^{max} = \max_{u \in M}(\omega(u)) < \infty \quad \textit{and} \quad \omega^{min} = \min_{u \in M}(\omega(u)) < \infty \tag{3.7}$$

The cardinality is defned by

$$\psi : M \longrightarrow \mathbb{R}_+, \quad \psi(M) := n < \infty. \tag{3.8}$$

The cardinality is always finite because we can't have an infinite amount of user in a real context. It can be seen as a measure of usage of the contract.

Next we summarize some useful properties.

Lemma 3.9. *Let $(M, \mathfrak{T}(M), \mathfrak{p}(M))$ be a society, $u \in M$ a user and $r \in \mathbb{R}_+$ a rate. Then we have following properties:*

(i) *$\pi_M(r)$ is a surjectiv map.*

(ii) *We have: $\pi^b_M(r_2) \subseteq \pi^b_M(r_1)$ and $\pi^l_M(r_1) \subseteq \pi^l_M(r_2)$ for $r_1 \leqslant r_2$.*

(ii) $\pi^l_M(\omega^{max}_l) = \mathbb{1}_{M_l} \quad \textit{and} \quad \pi^b_M(0) = \mathbb{1}_{M_b}$

(vi) *$\psi(\pi^l_M(r) \bigcup \pi^b_M(r)) = \psi(\pi^l_M(r)) + \psi(\pi^b_M(r))$ for all $r \in \mathbb{R}_+$.*
$\psi(\pi^l_M(r) \bigcup \pi^b_M(r)) = 0$ if and only if $\omega^{min}_l > r > \omega^{max}_b$.

Proof. Because $\pi_M(r)$ is a projection it is surjectiv by definition. This shows (1).

We set $r_1 \leqslant r_2$. Then $\pi^l_M(r_1) \subseteq \pi^L_M(r_2)$ follows directly from the Definition 3.8 as well as $\pi^b_M(r_2) \subseteq \pi^b_M(r_1)$. This proofs (2).

Since, the society is finite with $w_l(r) < \infty$ by Definition 2.4 there exists a $\omega^{max}_l < \infty$. Using (2) we end up with (3). Because, zero is the smallest possible wish rate and again using (2) gives the second relation of (3).

This follows directly from the definition of the cardinality in 3.8 and

$$\psi(\pi^l_M(r) \bigcup \pi^b_M(r)) = m = n_l + n_b = \psi(\pi^l_M(r)) + \psi(\pi^b_M(r)).$$

The second relation only holds when $\pi^l_M(r) = \varnothing$ and $\pi^b_M(r) = \varnothing$ holds. On the one hand we start with $\pi^l_M(r) = \varnothing$. This implies that $r < \omega^{min}_l$. For $\pi^b_M(r) = \varnothing$ one needs $\omega^{max}_b < r$ which gives the right result. On the other hand $\pi^b_M(r) = \varnothing$ implies again that $\omega^{max}_b < r$. Since $\pi^b_M(r) = \varnothing$ results in $r < \omega^{min}_l$ this ends the proof. □

As a result from Lemma 3.9 we clearly don't want a society in which $\omega^{min}_l > \omega^{max}_b$ occurs. Nevertheless, this is not in the power of the contract developer. It nicely shows that efficient lending and borrowing is only possible when the users are willing to pay a higher borrow APY than lending APY. This can also be seen as a good test for he chosen definitions. We can also read from the above lemma that, depending on the state of the society and rate, one gets a higher or lower usage. The sweet spot

for R lies in between ω_b^{max} and ω_l^{min} and clearly is not static at all! Next, we define abstractly what someone can understand about increasing the efficiency of a lending protocol.

Definition 3.10. *Let $(M, \mathfrak{T}(M), \mathfrak{p}(M))$ be a society and $R_p : \mathbb{R}_+ \longrightarrow \mathbb{R}_+$ a certain bonding curve out of the family of bonding curves $(R_p)_p$. We will write $R_p \equiv R(p)$. To increase the capital efficiency of a lending smart contract, one wants to maximize the number of user over time, meaning to find the best bonding curve for a given society. Therefore, we will see the pole p as a function of time t with $p(t) \in [p_{min}, p_{max}]$, $\forall\, t$. In other words:*

$$\forall t \ : \ \max_{p(t)\,\in\,[p_{min},\,p_{max}]} \left\{ \psi\big(\pi_M^l(R(p(t))) \bigcup \pi_M^b(R(p(t)))\big) \right\} \tag{3.9}$$

Now we have everything together to make following statement:

Theorem 3.11. *In the long time mean, meaning that $t \gg t_0$ where t_0 is the starting time, Definition 3.10 can be expressed equivalently in terms of maximizing S_l.*

Proof. By Lemma 3.9 (4) we can write for a fixed t

$$\max_{p(t)\,\in\,\mathbb{R}_+} \left\{ \psi\big(\pi_M^l(R(p(t))) \bigcup \pi_M^b(R(p(t)))\big) \right\} = \tag{3.10}$$

$$\max_{p(t)\,\in\,\mathbb{R}_+} \left\{ \psi\big(\pi_M^l(R(p(t)))\big) + \big(\pi_M^b(R(p(t)))\big) \right\} \leqslant n_l^{max}(t) + n_b^{max}(t)$$

This means the maximal efficiency in a given state of a society is at least bounded by a certain number of borrowers and lenders for a time t. By looking at the long time mean, we have $t \gg t_0$ and thus we can apply a mean value for the lending amounts μ_l and borrow amounts μ_b.

Following, we have a maximal amount from all lender which is $A_l^{max} = n_l^{max}(t)\mu_l$. This is also an upper bound for the maximal borrow amount, meaning $A_b^{max} = n_b^{max}(t)\mu_b \leqslant A_l^{max}$. Additionally, we can write

$$n_l^{max}(t) + n_b^{max}(t) \overbrace{\longrightarrow}^{t \gg t_0} \frac{A_l^{max}}{\mu_l} + \frac{A_b^{max}}{\mu_b} \leqslant A_l^{max}\left(\frac{1}{\mu_l} + \frac{1}{\mu_b}\right) = cA_l^{max} \tag{3.11}$$

So, in the long time limit, the maximum is bounded by A_l^{max}. This value can be directly converted into lending shares by an isomorphism. This ends the proof. □

One may also show that this is equivalent to maximize S_b. We want to highlight that this needs to be seen as a long time mean. In this case, inorganic spikes caused by whales or third part manipulations are canceled out. Only, in this scenario, the lending system is optimized by the society, and not by a malicious third entity.

3.2. **The LASA Algorithm.** The previous sections lay out the framework in which we can define the LASA algorithm. Its main objective is to maximize S_l of the contract, which is motivated by Theorem 3.11. LASA is an algorithm which tries to improve the outcome of the contract by varying one parameter. This makes LASA a one dimensional monte carlo method, which can be understood as an simple A.I.

The parameter which will be changed by LASA is the pole value p. Because we work in solidity, the major task is to keep the algorithm as simple as possible. The next diagram shows the flowchart of LASA **??**. Afterwards, we will explain the different steps and variables in more detail. The implementation can be found, as an example,

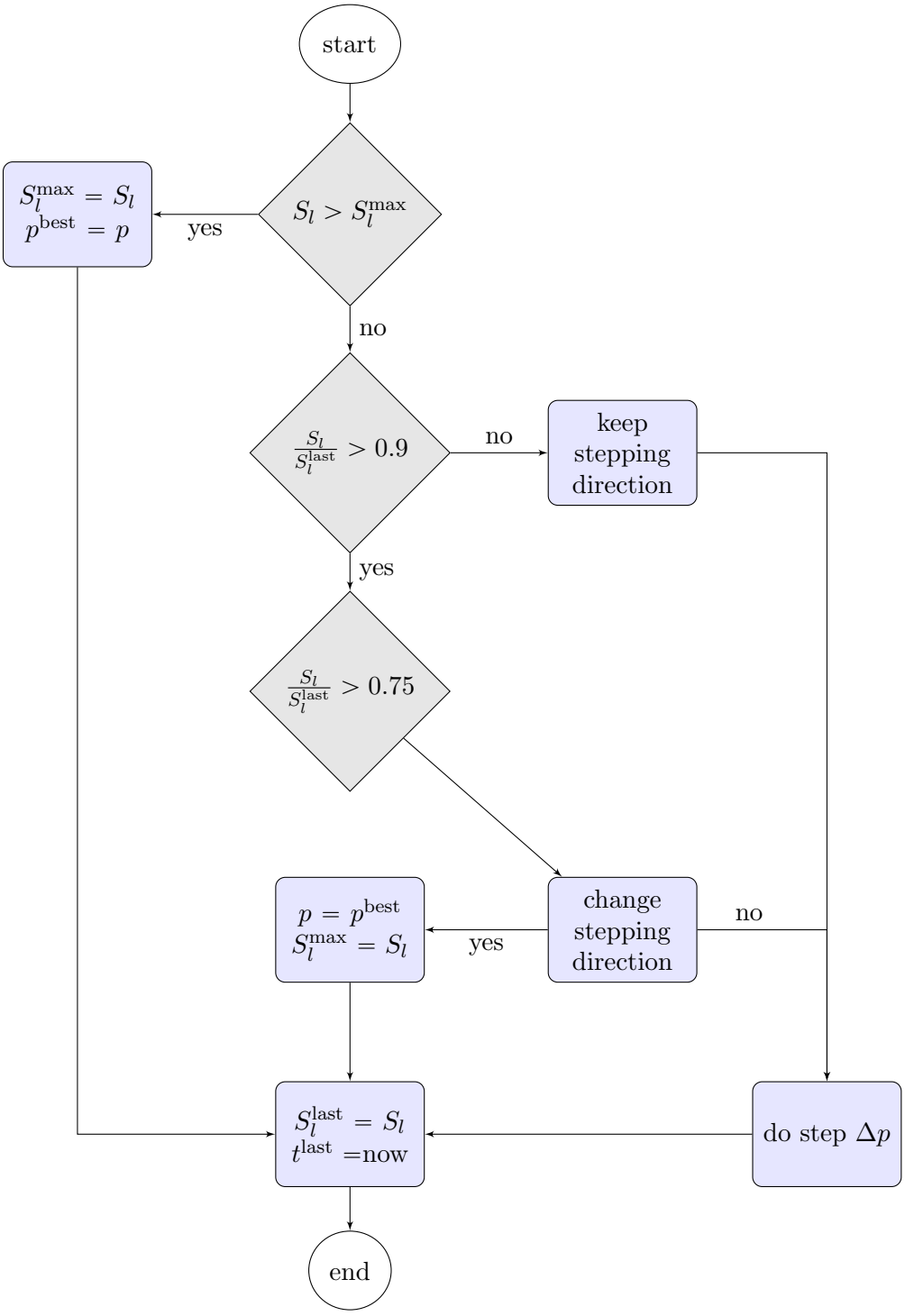

FIGURE 5. Flowchart for LASA.

in the WISE repository when published[2]. The variables in the above Flowchart **??** standing for:

- $S_l^{\max}$: The all time maximal value of lending shares inside the contract.
- p^{best}: Corresponding pole value for the $S_l^{\max}$.
- t^{last}: Time of last activity of LASA.
- S_l^{last}: Lending shares from the last activity round of LASA.

In general, the algorithm is probing the reaction from users, which spurs a change in the current borrow rate by comparing total share values. Doing so, it increases or decreases the pole value by a static stepping size Δp. At the beginning of each round, it compares the current values with the saved ones from last round. If a certain threshold is reached it is doing on of the three processes in the above Flowchart **??**.

- If the new share value is a new maximum, the algorithm saves the new max value and the corresponding pole value because a new global maximum in S_l is a positive sign in efficiency of the state. The pole values stays the same.
- In contrast, when S_l is reduced more than 10%, the algorithm changes the current stepping direction because S_l decreases and this is a negative feedback. We chose a tolerance of 10% to deal with daily fluctuations in the market and in the interest of not being over sensitive. When the change is inside this tolerance, the direction

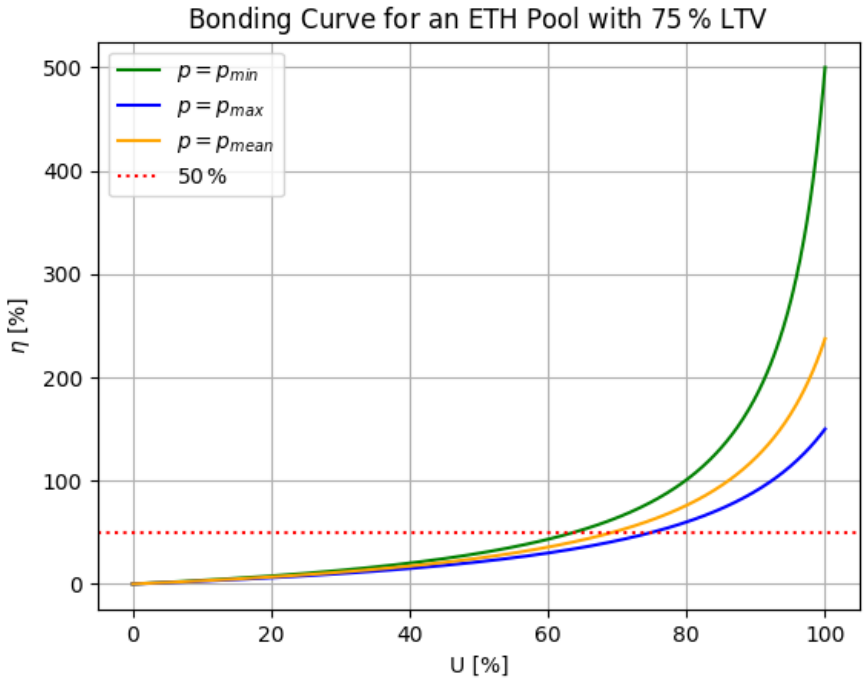

FIGURE 6. Plot of $R(x)$ for different p values.

stays the same. Afterwards, the algorithm performs a step.

- When the share amount reduces more than 25% compared to last the time, the algorithm again changes the stepping direction and sets the pool factor to the last known best value. This can be seen as a hard reset when big market changes occurred. No step is done.

The algorithm itself can only change p within its boundaries p_{min} and p_{max}. Even when a reset is triggered falsely by a whale, LASA will again find the best values since it constantly checks for user behavior. Furthermore, the difference in the borrow rate curve between p_{min} and p_{max} for the same utilization is not too large - An example can be seen in Figure (6). The effect growths with increasing utilization, leaving the small borrow rates nearly identical. The algorithm tries to improve the capital efficiency at higher utilizations because the acceptable borrow rates are bounded by ω_b^{max}. These rates are mostly satisfied at lower utilizations, highlight that LASA can move the curve from Figure (6) in the whole range between the blue and the green curve.

4. Numerical Simulations

4.1. **Simulation Setup.** One way to test the functionality of LASA could be locally with with truffle and JavaScript. But simulating a whole life-cycle within a JavaScript step by step is very exhausting and not efficient. Thus, we chose the C++ programming language to simulate a society, since it fit our needs the best.

Doing so, we firstly wrote a pendant of a slimed lending contract in C++. It included a basic share system for lending and borrowing. The user is limited to borrow/payback and lend/withdraw functions. Secondly, we implement the LASA algorithm with the corresponding borrow rate Function 3.4 as we would do inside a smart contract in solidity. Additionally, we implemented the borrow curve from Aave to mimic a second Aave contract which we used to produce reference values.
Then we developed a "fake" blockchain algorithm which managed the user interaction with the contract. The goal was to keep it as close as possible how user would interact on a real blockchain. The code can be found in the WiseSoft LLC repository when published [2]. In the following we will sketch the functionality:

At the beginning of each life-cycle the algorithm created a society. Then it saved the corresponding borrower and lender in separated vectors. Each user u was modelled by an object with fields describing their traits ξ, η, amount x and interaction time δ. By interaction time, we mean a maximal time a user lends or borrows even if their wish rate is still satisfied. This accounts for the fact that users sometimes shift their investment due to other reasons. The distribution is again of the gaussian type.
The length of each round withing a life-cycle was set as a day, and the time between the interactions defined by

$$(n_b + n_l)\Delta t = 86400.$$

After each round, we mixed up the order in both arrays, in order to minimize the possibility of having the same sequence of user interacting with the contract twice. A flowchart of the interaction algorithm can be seen below in Figure (**??**).

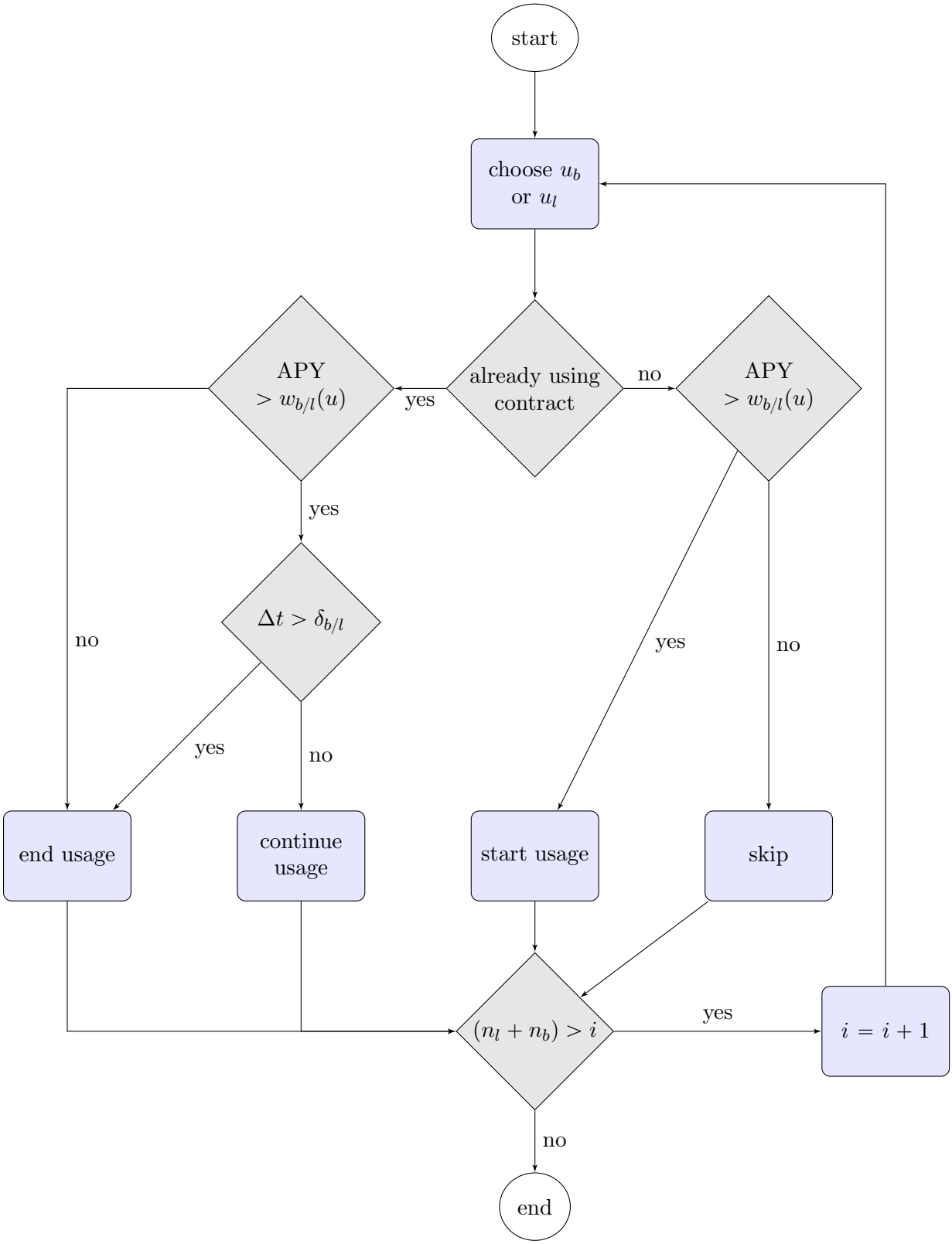

FIGURE 7. Flowchart of the blockchain simulation algorithm for a "day".

- It randomly chooses between a user or a lender by generating a random number between zero and one.
- Afterwards, it checks if the user is already borrowing or lending.
- If one is the case, it checks if the wish APY is still satisfied and if the maximum interaction time has not been reached.
- If this is the case, the algorithm lets stay the users in the contract. If not, it lets the user leave the contract.
- If the user is new, it checks if the lending or borrowing conditions for joining are fulfilled.
- These steps are repeated until all user interacted once. This corresponds to a time-frame of one day.
- This is repeated N-times, denoting a full life-cycle - normally with length of a year.

After each day, we saved the contract state variables like S_l, U, ξ, η etc. In the end, these have been evaluated with standard statistical methods. We calculated the average of these variables and determined their standard deviation to have a measure for their quality. The results are summarized in the next subsection.

4.2. **Simulation Results.** We begin by explaining the chosen parameters for the simulations in order to make our results reproducible. Afterwards, we show plots which indicate the difference and benefits of LASA compared to the standard method from Aave.

We decided to simulate a year of usage for a randomly generated society. This model has been repeated 1000 times. In other words, we compared the Aave method with LASA for 1000 different societies over a year each. Then we computed the mean values over these 1000 iterations for S_l, ξ and U. Each round by itself is statistically unrelated to another because they each contain randomly generated societies. But, when looking at the mean of a hugh sample size a trend can be seen. We decided against using a moving average, because then one can better see, that LASA tries to optimize the usage for each society, which is reflected by a smaller width of possible outcomes. To smoothing these 1000 points and approximate them with a curve, we used a *savgol-filter* method with a *window size of* 51 and *polynomial order of five*.

We simulated an ETH-Pool and set the parameter for the Aave implementation as described in [1]. The formula is defined like

$$R_{aave}(t) = \begin{cases} R_0 + \dfrac{U(t)}{U_{opt}} R_1 & \text{for} \quad U(t) < U_{opt} \\ R_0 + R_1 + \dfrac{U(t) - U_{opt}}{1 - U_{opt}} R_2 & \text{for} \quad U(t) \geqslant U_{opt} \end{cases} \tag{4.1}$$

with the parameters

$$R_0 = 0, \qquad R_1 = 4\,\%, \qquad R_2 = 300\,\%, \quad \text{and} \quad U_{opt} = 45\,\%.$$

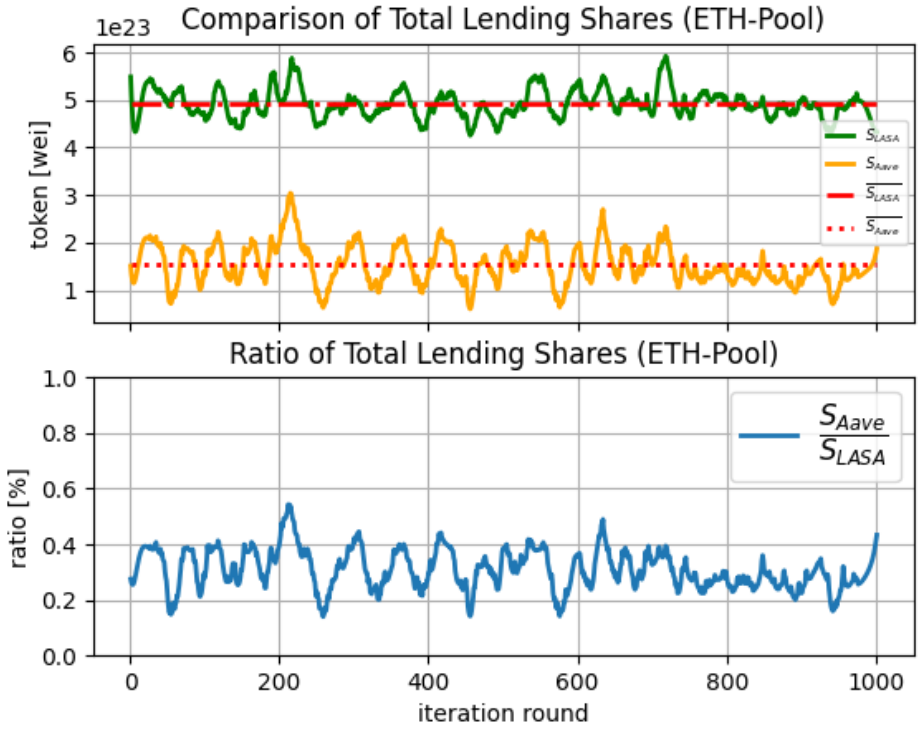

FIGURE 8. Plot of shares for different models. An ETH-Pool was simulated with parameters above since 4.2.

Additionally, the parameter for the simulation, and thus for the society and LASA has been chosen as:

- Iteration rounds: 1000
- Duration per iteration: 12 months
- Mean value μ_ξ: 0.83 %
- Mean value μ_η: 1.43 %
- Standard deviation lending rate σ_ξ: 0.25
- Standard deviation borrow rate σ_η: 0.25
- Multiplication factor a: 0.01

We defined the mean values by looking at the total mean borrow and lending APY at Aave from inception.

In Figure 8 we see plots for total share values from the simulations with their mean values. The orange one is from the Aave model, the green is from LASA and the

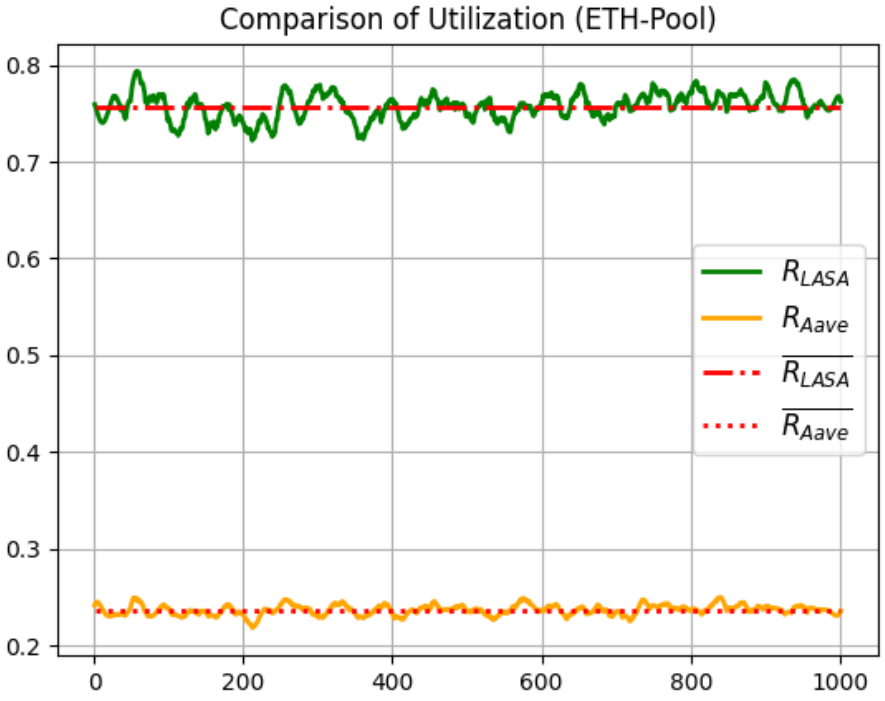

FIGURE 9. Plot of the utilization for different models. An ETH-Pool was simulated with parameters above from 4.2.

blue one is the quotient of both. One can see clearly that the overall share value of our LASA model ich much greater compared to the Aave one. The spikes arise from the fact that each point is its own iteration of the simulation, showing the total share amount after simulating a full year.

In the second Graphic 9 one can see the difference in utilization for both models. Again, the green plot is from LASA, the orange from Aave. With the chosen multiplication factor and the work of, LASA the utilization is much higher compared to Aave. Since the slops and U_{opt} of Aave is static, the system gets stuck with the given society for each attempt. Therefore, the changes in the utilization will not defer much between different iterations. On the contrast LASA tries to push the system in the direction, that it gets a positive feedback from the society. This works for some simulation rounds - thus for different societies - better than for others, resulting in a higher difference between each simulation. But, overall the mean is much higher compared to Aave.

In the last Plot 10 we compared the lending APYs of both systems. On can see that for most of the values the LASA algorithm reproduces better lending APYs than Aave. Thus, the mean value is higher. Additionally, a second effect of LASA can be seen in that the algorithm tries to maximize the efficiency of the contract by smoothing the curve. Every new society is generated randomly within the same distribution, which means they have the same mean values. The goal of the algorithm is to find this

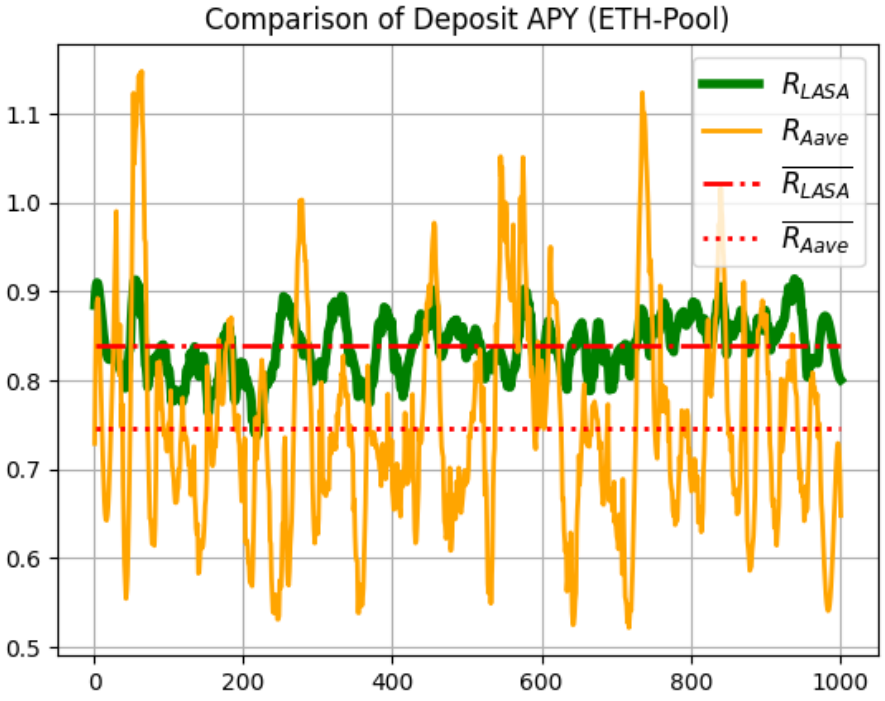

FIGURE 10. Plot of the deposit APY for different models. An ETH-Pool was simulated with parameters above from 4.2.

mean value, because with this, most of the users are satisfied, which results in the maximal number of lending shares. When one compares the mean value of the plot, it is nearly equal to the value of the society μ_ξ: 0.83 %. Due to random effects from the simulation and the randomly distributed usage time δ this happens in some life-cycles better compared to others. Overall, the standard deviation is small compared to Aave. Since the Aave system has static slopes with a static U_{opt}, the system can sometimes result in a good response for a society with high ξ. But in all other cases, the capital is not used efficient. One can argue that these spikes are good enough, and user can make good money when they "ride" these spikes, but it is very hard to predict them. Also, only a few benefits come from this. LASA doesn't have these spikes, but as shown overall, a static higher APY ξ which reflects higher capital efficiency.

5. Conclusion

To summarize, we have developed a new implementation for the bonding curve used in decentralized lending systems - which is also usable in NFT lending platforms. This method uses a asymptotic functions with a pole at p to reproduce the rapid increase at high utilizations. This pole factor can be seen equivalently as an U_{opt}. We showed with simple mathematical steps that $R(x)$ is a well-defined and smooth function.

Additionally, we developed a simple mathematical framework in which one can define the efficiency of a lending protocol. The idea is to set a seed for further and more complex investigations into these contracts, which may can be used in other

classes of contracts as well. Moreover, with the definition of a society, a basic setup for numerical simulations of smart contracts was developed.

Within this mathematical framework, an algorithm was developed (LASA), with the intention of increasing capital efficiency in lending protocols. With numerical simulations, the efficiency of LASA was tested and discussed. We have proven that, compared to other systems like Aave, LASA has better capital efficiency for an ETH pool. Additionally, the lending APY is in the mean higher compared to Aave.
Again, we want to highlight one strength of LASA. When looking at the Figure 10 we see the different APY over 1000 iterations. Each iteration has its own society, which also can be seen as different macro economic conditions. For each setting, LASA manged to keep the result very close to the mean value of the society. This can be interpreted as an equal lending APY over all market conditions. - e.g. compare a bull market versus a bear market. In contrast, one can't see spikes in the APY, which is the case for the Aave systems. Since, in some market conditions plus given randomness the Aave system may have a positive effect on the society but in reality, these are very rare and only a few can profit from them. Therefore, LASA also helps the flatten the APY range, which helps to distribute lending rewards to a broader mass continuously.

Of course, these results are merely our interpretations of simulations. In the end, the efficiency will show in real time applications. But for now, the results look quite promising. In the future, we want to expand on LASA's mathematical framework in follow-up papers. This was only a first set up with some basic definitions. We are curious what a deeper look into the mathematical modeling of smart contract models will bring.

References

[1] Aave Documentation (Gitbook), Version3, https://docs.aave.com/risk/liquidity-risk/borrow-interest-rate

[2] WiseSoft, LLC Repository, Github, https://github.com/wise-foundation

† Fakultät für Mathematik, Universität Regensburg, D-93040 Regensburg, Germany
Email address: christoph.krpoun@mathematik.ur.de
Email address: rene.hochmuth87@gmail.com